Probabilistic Model Building GP with Belief Propagation

Hiroyuki Sato*, Yoshihiko Hasegawa†, Danushka Bollegala‡ and Hitoshi Iba*
*Department of Electrical Engineering and Information Systems
Graduate School of Engineering The University of Tokyo, Tokyo, Japan 113-8654
† Department of Biophysics and Biochemistry,
Graduate School of Science, The University of Tokyo, Tokyo, Japan 113-0033
‡ Department of Electronics and Information,
Graduate School of Information Sciences, The University of Tokyo, Tokyo, Japan 113-8654

***Abstract*—Estimation distribution algorithms (EDAs) which deal with tree structures as GP are called as probabilistic model building GPs (PMBGPs), and they show better search performance than GP in many problems. A problem of prototype tree-based method, a type of PMBGPs, is that samplings do not always generate the most probable solution, which is the individual with the highest probability and reflects a learned distribution most. This problem wastes a part of learning and increases the number of evaluations to get an optimum solution. In order to overcome this difficulty, this paper proposes a hybrid approach using Belief propagation (BP) in sampling process. BP is an inference algorithm on graphical models and can generate the most probable solution. By applying our approach to benchmark tests, we show that the proposed method is more effective than PLS alone.**

I. Introduction

In this paper, we introduce loopy belief propagation (LBP) in probabilistic model building GPs (PMBGPs) in order to generate the most probable solutions in sampling process. We selected program optimization with linkage estimation (POLE) [1] as a foundation of our approach, which employs Bayesian networks for a probabilistic model and uses a special chromosome called as the expanded parse tree (EPT) [2]. We call our proposed method as POLE-BP.

Estimation of distribution algorithms (EDAs) are a new paradigm in the field of evolutionary computation, and have attracted more and more attention due to their reliability in GA-hard deceptive problems. EDAs estimate dependencies between loci using probabilistic models and search solution candidates. From a viewpoint of estimation of probabilistic distribution, GA samples solutions from unknown superior distribution using genetic operators such as mutation and crossover. On the other hand, EDAs repeat model learning and sampling from the model by taking advantage of the assumption that superior distribution can be well approximated by parametric models, which is in contrast to sampling with the genetic operators inspired by the natural evolution. EDAs with Bayesian networks (Bayesian optimization algorithm (BOA) [3], estimation of Bayesian networks algorithm (EBNA) [4] and learning factorized distribution algorithm (LFDA) [5]) are typical EDAs, because Bayesian networks are able to represent dependencies between any loci, and they can solve many real world problems [6] having complex dependencies among variables.

The individual which reflects a learned probabilistic model most is the one having the highest joint probability, which is often referred to as the *most probable solution*. However, many sampling methods in EDAs such as probabilistic logic sampling (PLS) and Gibbs sampling do not always generate the most probable solution at each generation. Therefore, some hybrid approaches using belief propagation (BP) algorithms to generate the most probable solution were proposed in GA-type EDAs. BP is widely used for inference in graphical models, and is a general algorithm to calculate marginal probabilities or the configuration with the highest probability in graphical models. Although the original BP can be only applied to graphs without cycles, BP has been applied to many applications accompanied by cycles, in which case the BP is specifically called loopy belief propagation (LBP). LBP achieves fairly good successes with low computational cost, although the guarantee that BP always converges to the exact solution within finite steps in acyclic graphs is lost in LBP. Because Bayesian networks used in EDAs are inevitably accompanied by cycles, approximate inference by LBP is popular in the field of EDAs [7], [8] as well as other applications.

The idea of estimation of probabilistic distribution has been applied to tree structures conventionally dealt by GP (Genetic Programming). EDAs for tree structures assume that the promising solutions can be well described by probabilistic models. They learn models and estimate parameters repeatedly to generate new tree structures. These EDAs for tree structures are called Genetic Programming - EDAs (GP-EDAs) [9] or probabilistic model Building GP (PMBGPs) [10]. PMBGPs are superior to GP in the sense that PMBGPs can search solutions with smaller number of fitness evaluations and is able to solve problems which conventional GP cannot solve. PMBGPs are broadly classified into two types. One type uses probabilistic context free grammar (PCFG) and learns production rule probabilities. The other type is prototype tree-based method, which converts trees to 1 dimensional arrays and applies GA-type EDAs to them. From the viewpoint of probabilistic models, the prototype tree-method is essentially equivalent to GA-type EDAs and hence it can easily incorporate techniques devised in the field of EDAs.

In the present paper, we employ LBP in the prototype tree-based method to sample the most probable solutions in probabilistic models. At each generation, trees are converted to 1 dimensional arrays, and dependencies between variables are estimated, using Bayesian networks. Because LBP has to run on a factor graph, which is a kind of graphical models and an undirected graph, a learned Bayesian network is transformed to an equivalent factor graph. LBP on the factor graph generates the individual with the highest joint probability, and it is incorporated into the next generation in addition to individuals sampled from Bayesian networks. The superiority of the proposed method is shown with three benchmark tests.

The rest of the paper is organized as follows. Section 2 and 3 briefly introduce PMBGPs and LBP algorithms, respectively. Section 4 explains details of the proposed method. Section 5 presents the experimental condition and results, which is followed by the discussion in Section 6. Finally Section 7 concludes the paper.

II. Probabilistic Model Building GP

Because PMBGPs construct probabilistic models on tree structures, they can express dependencies between nodes more flexibly than GP. While GP propagates building blocks by crossover and dependencies expressed by building blocks are limited to sub trees, PMBGPs can express dependencies between brothers or ancestors and descendants besides sub trees. Two types of methods, a prototype tree-based method and a PCFG-based method, are known in the field of PMBGPs. The prototype tree-based method translates trees to 1 dimensional arrays and applies GA-type EDAs to them. On the other hand, PCFG-based method expresses individuals with derivation trees, and learns production rules as well as their parameters [11]–[15]. Since recent PCFG-based methods weaken the context freedom assumption (probabilities of production rules do not depend on parent or sibling nodes) by using more advanced PCFG models, they can estimate position independent building blocks [16]. Despite of these facts, the prototype tree-based methods are advantageous over PCFG-based methods in the sense that they can easily utilize existing GA-type EDAs, and consequently, state-of-the art in GA-type EDAs [17], [18] can be directly introduced to PMBGPs.

A. Prototype tree-based method

Prototype trees are full α-ary trees, where α is the maximum number of arguments among function nodes. Nodes of prototype trees are numbered in breadth-first order, and trees are converted to 1 dimensional arrays according to the order. Because all the individuals have the same dimension (the same number of nodes) due to individuals being perfect trees, GA-type EDAs can be applied to the arrays. Probabilistic incremental program evolution (PIPE) [19] assumes that every node is independent of each other, which can be considered as an adaptation of population-based incremental learning (PBIL) [20] to GP. Estimation of distribution programming (EDP) [21] takes into account the parent-children relationships in tree structures. Extended compact GP (ECGP) [22] combines the extended compact GA (ECGA) [23], which estimates the group of marginal distributions using the minimum description length (MDL) principle, with GP to take into account the multivariate dependencies among nodes. Bayesian optimization algorithm (BOA) programming (BOAP) [24] is based on BOA and uses a zig-zag tree to represent functions and programs. Program optimization with linkage estimation (POLE) also estimates the dependencies among nodes by Bayesian networks.

The PMBGPs listed above use PLS to generate new individuals. However, with PLS, there is no guarantee that the most probable solutions, which have the highest joint probability and reflect learned probabilistic models the most, are generated at each generation. This is the key issue especially for the case of small population size. In order to overcome the deficiency, we introduce LBP to PMBGPs in the sampling process, and the most probable solutions are generated at each generation with LBP. We employ POLE as a baseline PMBGP in our study, which will be briefly explained in the next section for readers convenience.

B. Program Optimization with Linkage Estimation

POLE is a PMBGP using Bayesian networks and EPT for expressing tree structures. Although prototype tree-based PMBGPs can incorporate techniques devised in GA-type EDAs, there are problems in the naive application of GA-type EDA techniques to GP. Because GP uses more symbols (e.g., functions and terminals) than GA (e.g. 0 and 1), the number of possible symbols at each position is very large. Since the number of possible symbols is the base of exponential increase in conditional probability table (CPT) size, a use of more symbols results in more population size in PMBGPs. Furthermore, all the programs sampled from Bayesian networks must satisfy syntactic correctness, which can not be ensured in general.

POLE translates programs to perfect trees, using EPT which makes all function nodes take α arguments (α is the maximum number of arguments among function nodes). The selection node L is the function node defined as $L(x, y, ...) = x$, which only evaluates its first argument and return the result.

EPT solves two problems of prototype tree-based methods listed above. A use of EPT guarantees the syntactic correctness and ameliorates the difficulty in estimating dependencies due to increase of symbols.

EPT moves nodes on trunk to leafs and generates equivalent programs with perfect trees, using L. Figure 1 shows an example of EPT along with its equivalent standard GP tree expression. The EPT inserts selector operators L into the standard GP tree in order to ensure that terminal nodes are positioned at the leaves of the full α-ary tree.

Let F and T be sets of function and terminal nodes, respectively. Symbols which each node can take are given by $F \cup T$ for the case of normal trees. EPT reduces trunk symbols to $F \cup \{L\}$ and terminate symbols to T. Consequently, model complexity reduced by few symbols enables fast parameter learning and the reduction of the number of evaluations. At the same time, all the programs generated by Bayesian networks under EPT are syntactically correct, which is not the case for the standard GP tree case. The details of POLE are described

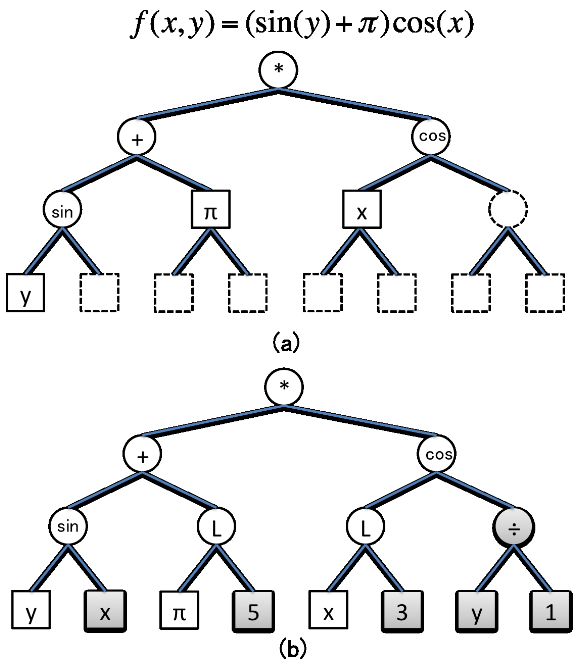

Fig. 1. (a) standard GP tree and (b) EPT. The grey nodes are introns. Circles are function nodes and squares are terminate nodes. (a) and (b) are syntactically identical.

below.

Step 1 Initialization of individuals
Trees are initialized by Grow.

Step 2 Evaluation of individuals

Step 3 Selection of individuals
According to a truncate selection, individuals are selected which are used for construction of Bayesian networks and estimation of parameters. The number of individuals to select is represented with MP_s, where P_s is a selection rate and M is a population size.

Step 4 Construction of Bayesian network
On the basis of K2 algorithm [25], Bayesian networks are constructed with the samples and parameters are also estimated. Bayesian information criteria (BIC) [26] is used to evaluate networks. In our implementation, networks are constructed from scratch at each generation.

Step 5 Generation of new individuals
New M individuals are generated with constructed Bayesian networks by PLS. If we want to use the elitist strategy, better MP_e individuals are copied from previous generation, where P_e denotes elite rate.

PMBGPs except POLE simply translate trees to 1 dimensional arrays with fixed size and cannot avoid problems accompanied with increasing symbols. POLE solves these problems and performs better on deceptive MAX problem and Royal tree problem. Moreover, a extension of POLE was proposed in [27].

III. Loopy Belief Propagation

Original belief propagation (BP) infers marginal probabilities and instances of each node on Bayesian networks [28]. In BP, nodes repeatedly send *messages*, which are parts of locally calculated marginal probabilities, each other and update them in order to get marginal probabilities. This process is called as *Message Passing* in BP. Nowadays, BP is applied to calculation of joint probabilities as well as marginal probabilities and is expanded for Markov random field (MRF) and a factor graph. The algorithm for joint probabilities is called as max-sum or max-product, and that for marginal probabilities as max-product. BP on graphs with cycles has no guarantee to stop and converge to a correct solution while BP works correctly within finite steps on acyclic graphs. However, in practice, BP on graphs with cycles, which is called as LBP, works well in many applications [29], [30].

A. Loopy max sum

The proposed method applies max-sum to cyclic factor graph and generates the most probable solutions. We call this procedure as loopy max-sum. Let $\mu_{f\to x}$ and $\mu_{x\to f}$ be messages from factor nodes to variable nodes and those from variable nodes to factor nodes, respectively, and $ne(X)$ and $ne(X)\backslash Y$ be the set of adjacent nodes to X and the set of adjacent nodes to X except Y, respectively, and $f(x_0,\cdots,x_n)$ be the value of factor f when its adjacent variable nodes are $x_0,\cdots,x_n$. The details of loopy max-sum are described below.

Step 1 Initialization
All messages are initialized to 0.

Step 2 Message passing
Message passing is repeatedly executed, using Eqs. (1), (2), (3) and (4). Eqs. (1) and (2) represent messages from leaf nodes, and Eqs. (3) and (4) represent messages from nodes except leafs.

$$\mu_{f\to x}(x) = \ln f(x), \tag{1}$$

$$\mu_{x\to f}(x) = 0, \tag{2}$$

$$\mu_{f\to x}(x) = \max_{x_1,\cdots,x_M}\Bigg[\ln f(x,x_1,\cdots,x_M) + \sum_{m\in ne(f_s)\backslash x}\mu_{x_m\to f}(x_m)\Bigg], \tag{3}$$

$$\mu_{x\to f}(x) = \alpha_{xf} + \sum_{l\in ne(x)\backslash f}\mu_{f_l\to x}(x), \tag{4}$$

where α_{xf} is a scalar chosen such that

$$\sum_{x_n}\mu_{x\to f}(x) = 0.$$

Step 3 Check termination criteria
If termination criteria is satisfied, message passing ends. Otherwise, message passing continues.

Step 4 Get the most probable solution
Get the most probable solution, using Eq. (5).

$$x^{max} = \operatorname*{argmax}_{x}\left[\sum_{s \in ne(x)} \mu_{f_s \to x}(x)\right]. \qquad (5)$$

IV. The Proposed method

The proposed algorithm, POLE-BP, is different from the original POLE in the sampling process. POLE-BP generates one individual, the most probable solution, with loopy max-sum and the rest with PLS, in contrast to the original POLE where all the individuals are generated with PLS. Because loopy max-sum must run on factor graphs, Bayesian networks are translated to factor graphs. The details of POLE-BP is described below.

Step 1 Initialization (Identical to POLE)
Step 2 Evaluation of individuals (Identical to POLE)
Step 3 Selection of individuals (Identical to POLE)
Step 4 Construction of Bayesian network
(Identical to POLE)
Step 5 Generation of new individuals by sampling
$M-1$ individuals are sampled from the Bayesian network constructed by Step4.
Step 6 Graph translation
The Bayesian network is translated to the equivalent factor graph.
Step 7 Loopy max-sum
Loopy max-sum runs on the factor graph. The most probable solution gotten by Eq. (5) is carried to the next generation.

POLE-BP guarantees that population includes the most probable solution, which is an individual having the highest joint probability, in each generation. We expect that POLE-BP performs better than conventional POLE especially in small population cases where it is hard to generate the most probable solution with PLS.

V. Experiments

In order to investigate the search performance of our approach, we applied POLE-BP to three problems: the MAX problem, the deceptive MAX (DMAX) problem and the royal tree problem. The main parameters in common with POLE are listed in Table I.

In the experiments, we compared the performance of the three models listed below

- POLE-BP
  LBP is incorporated into POLE. The main parameters are identical to POLE. Message passing schedule of loopy max-sum is that messages are sent from all factor nodes and variable nodes by turns. When all nodes send messages 100 times, termination criteria is met. This termination criteria is enough for convergence, because messages usually converge within about 10 passing in our observation.
- POLE
  To estimate interactions between nodes, a Bayesian network is constructed. The maximum number of incoming edges per node is unlimited.
- Simple GP (SGP)
  This algorithm is a simple implementation of GP. Let P_c, P_m and P_r be crossover rate, mutation rate and reproduction rate. In the experiments, we use $P_e = 0.005$, $P_c = 0.995$, $P_m = 0$ and $P_r = 0$. Crossover points are selected with bias. In applying the crossover operator to two individuals, we selected the first crossover point from function nodes with a probability of 0.9, and from terminals with a probability of 0.1. The second crossover point is selected under the condition that the depth of both individuals does not exceed the depth limitation. The tournament size is set to $T_s = 2$. Grow is used to initialize individuals.

The population size of each method is determined in the following way. We start from $M = 100$, and increase the population size by $\sqrt[5]{10}$ times. For each population size, we execute 20 runs. If the algorithm obtains the optimum solution 20 times from the 20 runs, we stop increasing the population size and calculate the average number of fitness evaluations. This method of determining the population size has previously been employed in [3], [4].

All algorithms are terminated when they obtain the optimum solution. Furthermore, we also adopted another termination criterion. Because POLE-BP and POLE tend to converge faster than SGP, these algorithms stop when the best fitness value at each generation is not improved for ten continuous generations. In contrast, SGP stops when the fitness values are not improved from generation g to generation $2g (g > 10)$.

We performed t-test (Welch, two-tailed) for each experiment. We calculate the P-value for the obtained data (e.g., the average of POLE-BP and the average of POLE) to exhibit any statistically significant differences between the results of the different approaches.

TABLE I
Parameters of POLE-BP and POLE.The missing concrete values (denoted by the " - " symbols) are problem dependent

Parameters	Meaning	value
M	Population size	-
D_p	Maximum tree depth	-
P_s	Selection Rate	0.1(MAX, DMAX) 0.2(Royal Tree)
R_p	Range of Parents	2(MAX, DMAX) 4(Royal Tree)
k	Maximum Incoming Edge	∞
P_e	Elitist reproduction Rate	0.05
P_F	Functional Selection Rate	0.8(MAX,DMAX) 0.9(Royal Tree)

A. Experiment 1: MAX Problem

1) Problem Description: The MAX problem [31], [32] is a benchmark test to examine the mechanism of GP crossover, and is widely used as a benchmark test for PMBGPs [15], [21]. A purpose of the MAX problem is to search a function that returns the largest real value within the limits of a maximum tree depth. In this problem, three symbols described below are used.

$$F = \{+, *\}, \quad T = \{0.5\} \tag{6}$$

An optimum solution can be obtained by the following procedure. First, create the value "2" using four "0.5" symbols and three "+" symbols. Then, multiply the created "2"s using "*" symbols. $2^{2^{D_p}-3}$ represents the optimum value for a given maximum depth D_p.

2) Results and Analysis: Let tree size be the number of nodes contained in the optimum structure, which can be represented by $2^{D_p} - 1$ in the MAX problem. Table II presents the average number of fitness evaluations and standard deviations in 20 trials. Fig. 2 visualizes Table II and describes the relationship of tree size versus the average number of fitness evaluations for the three models. Table III describes the results of the t-test, which indicates that POLE-BP is superior to POLE and SGP. According to Table III, the P-value for POLE-BP and POLE is smaller than 1% in $D_p = 6, 7, 8$. This means that difference between POLE-BP and POLE for $D_p = 6, 7, 8$ is statistically significant at 1% significant level. From the result above, LBP works better in deeper problems which requires larger population.

TABLE II
THE NUMBER OF FITNESS EVALUATIONS FOR THE MAX PROBLEM. VALUES OF THE STANDARD DEVIATION (STDEV) FOR EACH CASE ARE GIVEN IN PARENTHESES

		$D_p = 5$	$D_p = 6$	$D_p = 7$	$D_p = 8$
POLE-BP	average	380	1272	3475	10238
	stdev	(87)	(310)	(918)	(1567)
POLE	average	435	1648	4900	14679
	stdev	(85)	(263)	(307)	(1353)
SGP	average	1416	3208	8513	59950
	stdev	(249)	(442)	(1136)	(4801)

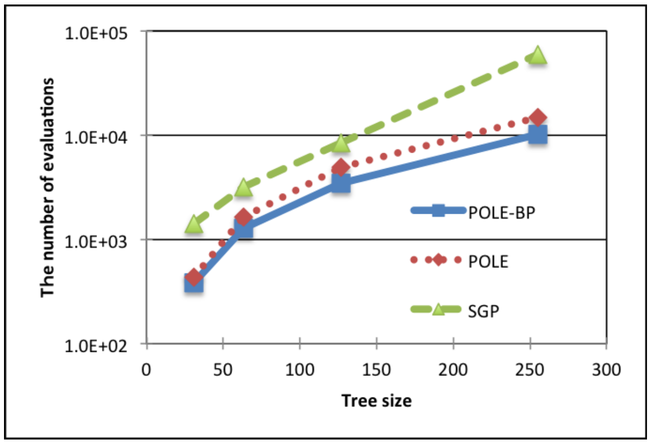

Fig. 2. The number of evaluations required for the MAX problem

TABLE III
t-TEST.THE VALUES REPRESENT P-VALUES FOR THE MAX PROBLEM

	$D_p = 5$	$D_p = 6$	$D_p = 7$	$D_p = 8$
POLE-BP vs POLE	5.02E-2	2.0E-4	1.02E-06	underflow
POLE-BP vs SGP	3.33E-15	underflow	underflow	underflow

B. Experiment 2: Deceptive MAX (DMAX) Problem

1) Problem Description: In order to investigate how LBP works in a deceptive problem, we also applied POLE-BP to a deceptive MAX problem (DMAX problem) [1], which is a deceptive extension of the MAX problem. The DMAX problem has the same objective as the MAX problem: to find functions which return the largest real value under the limitation of a maximum tree depth D_p.

$$F = \{add_5, multiply_5\} \tag{7}$$

$$T = \{\lambda_3, 0.95\} \tag{8}$$

$$add_5(a_0, \cdots, a_4) = \sum_{i=0}^{4} a_i \tag{9}$$

$$multiply_5(a_0, \cdots, a_4) = \prod_{i=0}^{4} a_i \tag{10}$$

$$\lambda_3 = \left(-\frac{1}{2} + \frac{i\sqrt{3}}{2}\right) \tag{11}$$

Let us consider the optimum value for the DMAX problem with $D_p = 3$. In order to get maximum absolute value, first, create $5\lambda_3$, using five λ_3 and add_5. Then, create $(5\lambda_3)^5 = 5^5{\lambda_3}^2$, using $5\lambda_3$ and $multiply_5$. However, $Re(5^5{\lambda_3}^2)$ is negative, and $5^5{\lambda_3}^2$ is not a optimum solution. Therefore, substituting two $5 * 0.95$ for two $5\lambda_3$ makes the optimum value, $(5\lambda_3)^3(0.95 * 5)^2 = 2820.3125$. Fig. 3 visualizes this structure. We can find that the optimum value with $D_p = 4$ is $(5\lambda_3)^{24}(0.95 * 5) = 2.83 * 10^{17}$ in a similar way.

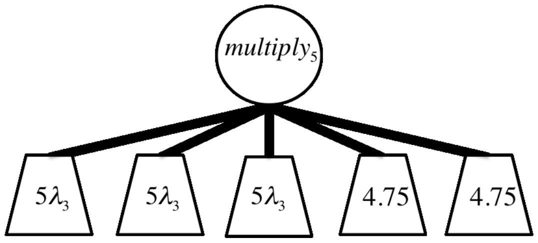

Fig. 3. An optimum solution of the DMAX problem ($D_p = 3$)

2) Results and Analysis: Table IV present the average number of fitness evaluations and standard deviations in 100 trials. SGP could not get an optimum solution at $D_p = 4$. Fig. 4 visualizes Table IV and describes the relationship of tree size versus the average number of fitness evaluations for the three models. Table V shows the result of t-test. Because each

P-value between POLE-BP and POLE is bigger than 1 %, the performance of POLE-BP may not surpass POLE.

We ran 100 different experiments where population is $M = 40, 50, 60, 70, 80, 90, 100, 160$ from others in order to investigate how LBP works in small population. Table VI presents the number of success trials of depth 3 in 100 trials. Fig. 5 visualizes effects of BP in small population. In $M = 40, ..., 100$, POLE-BP succeeded more frequently than POLE. In the DMAX problem, LBP works well in small population cases, but performs worse in large population cases in contrast to other two benchmark tests (the MAX and the royal tree problems).

TABLE IV
THE NUMBER OF FITNESS EVALUATIONS FOR THE DMAX PROBLEM. VALUES OF THE STANDARD DEVIATION (STDEV) FOR EACH CASE ARE GIVEN IN PARENTHESES

		$D_p = 3$	$D_p = 4$
POLE-BP	average	1539	120708
	stdev	(276)	(10569)
POLE	average	1517	122031
	stdev	(283)	(5819)
SGP	average	34875	-
	stdev	(4533)	(-)

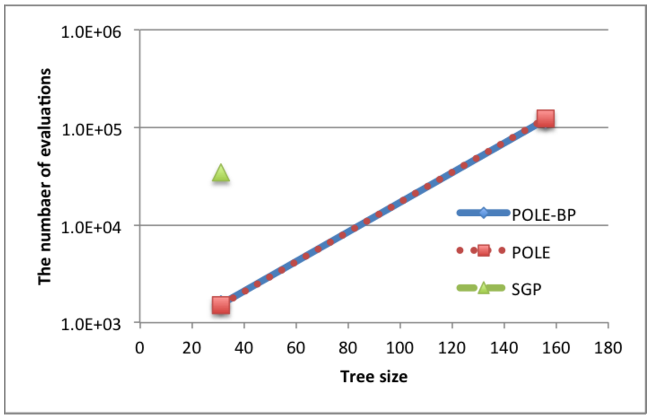

Fig. 4. The number of evaluations required for the DMAX problem

TABLE V
t-TEST.THE VALUES REPRESENT P-VALUES FOR THE DMAX PROBLEM

	$D_p = 3$	$D_p = 4$
POLE-BP vs POLE	5.78E-1	2.75E-1
POLE-BP vs SGP	underflow	-

C. Experiment 3: Royal Tree Problem

1) Problem Description: To show the effectiveness of POLE-BP, we applied our approach to the royal tree problem [33]. The royal tree problem is an extension of the royal road function [34], which is designed to examine the functionality of schema in GA, to GP. In the royal tree problem, GP gets

TABLE VI
THE NUMBER OF SUCCESS TRIALS IN 100 TRAILS (THE ROYAL TREE PROBLEM, $D_p = 3$)

	Population							
	40	50	60	70	80	90	100	160
POLE-BP	2	10	42	72	84	94	99	100
POLE	1	3	21	54	79	91	97	100
SGP	77	77	81	84	88	85	89	89

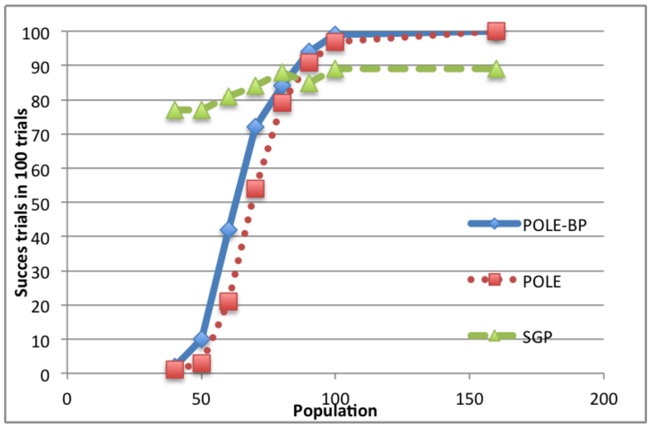

Fig. 5. The number of success trials in 100 trails ($D_p = 3$)

the optimum structure by combining the building blocks using crossover operators.

In this problem, symbols described below are used.

$$F = \{A, B, C, D, E, F\} \tag{12}$$
$$T = \{x\}, \quad x = 1 \tag{13}$$

An optimum solution is call as *Perfect Tree*, where each trunk node has the previous alphabet of children(e.g. children nodes of a function D are C). Each function node multiplies fitness of children by weight and adds them. If a tree which has children as root is perfect tree, the weight is Full Bonus. If a child is a correct node, but it is not a perfect tree, the weight is Partial Bonus. If a child is not correct node, the weight is Penalty. Moreover, if sub tree whose root is itself, fitness is multiplied by Complete Bonus. We use Full Bonus = 2, Partial Bonus = 1, Penalty = 1/3, Complete Bonus = 2. For example, Fig. 6 visualizes an optimum solution of $D_p = 4$, and its fitness is 512.

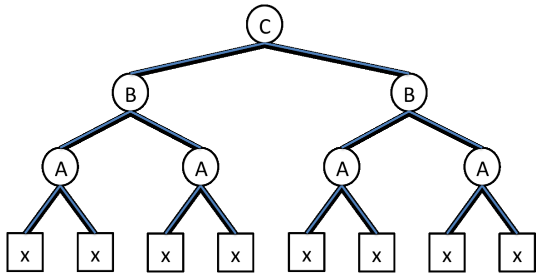

Fig. 6. An optimum solution of the Royal tree problem ($D_p = 4$)

2) Results and Analysis: Table VII presents the average number of fitness evaluations and standard deviations in 20 trials. Fig. 7 shows some of the data in Table VII and describes the relationship of tree size versus the average number of fitness evaluations for the three models. Table VIII represents the results of the t-test, which indicates that POLE-BP is superior to POLE and SGP. According to Table VIII, the P-value for POLE-BP and POLE is smaller than 1 % in $D_p = 6, 7$. This clearly shows that difference between the proposed method and POLE for $D_p = 6, 7$ is statistically significant at 1% significant level. As in the MAX problem, LBP works better in deeper problems which requires larger population size.

TABLE VII
THE NUMBER OF FITNESS EVALUATIONS FOR THE ROYAL TREE PROBLEM. VALUES OF THE STANDARD DEVIATION (STDEV) FOR EACH CASE ARE GIVEN IN PARENTHESES

		$D_p = 4$	$D_p = 5$	$D_p = 6$	$D_p = 7$
POLE-BP	average	260	1352	14720	67250
	stdev	(66)	(413	(2844)	(3250)
POLE	average	290	1560	25375	94600
	stdev	(54)	(469)	(1431)	(4432)
SGP	average	420	3000	29520	119800
	stdev	(144)	(316)	(2115)	(6750)

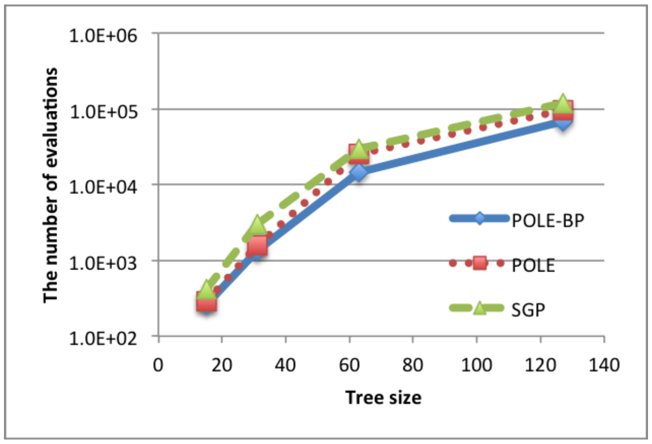

Fig. 7. The number of evaluations required for the Royal tree problem

TABLE VIII
t-TEST.THE VALUES REPRESENT P-VALUES FOR THE ROYAL TREE PROBLEM

	$D_p = 4$	$D_p = 5$	$D_p = 6$	$D_p = 7$
POLE-BP vs POLE	1.24E-1	1.45E-1	6.88E-15	underflow
POLE-BP vs SGP	1.11E-5	2.22E-16	underflow	underflow

VI. DISCUSSION

We have employed LBP in the sampling process of POLE in order to generate the most probable solutions. POLE, which estimates interactions between any nodes and overcomes problems accompanied with prototype tree-based methods, using EPT, is a powerful algorithm to solve problems which require estimation of dependencies between nodes. Nevertheless, POLE does not show good performance on small population cases due to the sampling bias with PLS. Regarding the sampling bias in PMBGPs, Ref. [35] recently reported that the sampling bias is of problematic especially for small population cases. Because our approach generates the most probable solutions with LBP, we have improved the success probability in small population cases in PMBGPs. We have applied the proposed method, POLE-BP, to three problems with different characteristics: the MAX problem (no interactions), the DMAX problem (lateral interactions) and the royal tree problem (parent—child interactions).

In the MAX problem, the number of the average fitness evaluations of the proposed method is superior to POLE and SGP. Because the MAX problem has no deceptiveness and the fitness monotonically increases by combining building blocks, Bayesian networks constructed in the MAX problem always grow toward optimal distribution. We consider that this would be a reason why generating the most probable solution works well in the problem. On the other hand, in the DMAX problem, the number of the average fitness evaluation between POLE-BP and POLE is not significant compared to the MAX problem case. DMAX problem has deceptiveness, and hence Bayesian networks do not necessarily converges to the optimum distribution as generation elapses. The most probable solutions generated from such distributions are not always superior, and LBP does not necessarily work well in deceptive problems. However, we found that POLE-BP tends to succeed better than POLE in small population cases. This results supports our expectation that in the small population cases, PLS does not work well due to the sampling bias, and generating the most probable solutions by LBP is effective. In the royal tree problem, it is significant that the number of the average fitness evaluation of the proposed method is superior to POLE and SGP. Because generating the most probable solutions by sampling is more difficult in royal tree problem which has more symbols than MAX and DMAX problems, effects of LBP have been shown more clearly.

VII. CONCLUSION

We have proposed POLE-BP, which is a hybrid approach which combines LBP and POLE, a PMBGP using prototype tree. Experiments for three benchmark tests showed that POLE-BP is superior to POLE alone.

In this paper, we applied POLE-BP to three benchmark tests in order to demonstrate effectiveness of POLE-BP. However, application to real world, especially multiobjective, problems is also important, and extension of our approach for such problems is a remaining objective.

ACKNOWLEDGMENT

This work was supported by a Grant-in-Aid for Scientific Research B (23119009) from MEXT (H.I.).

References

[1] Y. Hasegawa and H. Iba, "A Bayesian network approach to program generation," *IEEE Transactions on Evolutionary Computation*, vol. Vol. 12, NO. 6, pp. 750–764, 2008.

[2] M. Wineberg and F. Oppacher, "A representation scheme to perform program induction in a canonical genetic algorithm," *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature*, pp. 292 – 301, 1994.

[3] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz, "Boa: The Bayesian optimization algorithm," *IlliGAL Report No. 98013: Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL.*, 1999.

[4] R. Etxeberria and P. Larrañaga, "Global optimization using Bayesian networks," in *Proc. 2nd Symposium on Artificial Intelligence (CIMAF-99)*, 1999, pp. 332–339.

[5] H. Mühlenbein and T. Mahnig, "The factorized distribution algorithm for additively decomposed functions," 1999.

[6] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation (Genetic Algorithms and Evolutionary Computation)*. Kluwer Academic, 2002.

[7] R. S. A. Mendiburu and J. A. Lozano, "Introducing belief propagation in estimation of distribution algorithms: A parallel framework," *Technical Report. EHU-KAT-IK-11-07. University of the Basque Country*, 2007.

[8] C. F. Lima, M. Pelikan, F. G. Lobo, and D. E. Goldberg, "Loopy substructural local search for the Bayesian optimization algorithm," in *Proceedings of the Second International Workshop on Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, ser. SLS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 61–75. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-03751-1_5

[9] Y. Hasegawa and H. Iba, "Estimation of distribution algorithm based on probabilistic grammar with latent annotations," *Proceedings of IEEE Congress of Evolutionary Computation (CEC 2007)*, pp. 1043–1050, 2007.

[10] K. Sastry and D. E. Goldberg, "Probabilistic model building and competent genetic programming," *Genetic Programming Theory and Practise, chapter 13*, pp. 205–220, 2003.

[11] P. A. N. Bosman and E. D. de Jong, "Grammar transformations in an EDA for genetic programming," *GECCO 2004 Workshop Proceedings*, 2004.

[12] A. Ratle and M. Sebag, "Avoiding the bloat with probabilistic grammar-guided genetic programming," in *Artificial Evolution 5th International Conference, Evolution Artificielle, EA 2001*, ser. LNCS, P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, and M. Schoenauer, Eds., vol. 2310. Creusot, France: Springer Verlag, Oct. 29-31 2001, pp. 255–266. [Online]. Available: http://arxiv.org/PS_cache/cs/pdf/0602/0602022v1.pdf

[13] E. N. Regolin and A. T. R. Pozo, "Bayesian automatic programming," *Genetic Programming*, vol. vol. 3447, Lecture Notes in Computer Science, pp. 38–49, 2005.

[14] Y. Shan, R. I. Mckay, H. A. Abbass, and D. Essam, "Program evolution with explicit learning: a new framework for program automatic synthesis," *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pp. 1639–1646, 2003.

[15] Y. Shan, R. I. Mckay, and R. Baxter, "Grammar model-based program evolution," *In Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, pp. 478–485, 2004.

[16] Y. Hasegawa and H. Iba, "Latent variable model for estimation of distribution algorithm based on a probabilistic context-free grammar," *Trans. Evol. Comp*, vol. 13, pp. 858–878, August 2009. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2009.2015574

[17] R. Santana, P. Larrañaga, and J. A. Lozano, "Learning factorizations in estimation of distribution algorithms using affinity propagation," *Evol. Comput.*, vol. 18, pp. 515–546, December 2010. [Online]. Available: http://dx.doi.org/10.1162/EVCO_a_00002

[18] X. Li, B. Li, S. Mabu, and K. Hirasawa, "A novel estimation of distribution algorithm using graph-based chromosome representation and reinforcement learning." in *IEEE Congress on Evolutionary Computation'11*, 2011, pp. 37–44.

[19] R. P. Sałustowicz and J. Schmidhuber, "Probabilistic incremental program evolution," *Evolutionary Computation*, vol. 5, pp. 123–141, 1997.

[20] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Tech. Rep., 1994.

[21] K. Yanai and H. Iba, "Estimation of distribution programming based on Bayesian network," *Proceedings of the Congress on Evolutionary Computation 2003*, pp. 1618–1625, 2003.

[22] K. Sastry and D. E. Goldberg, "Probabilistic model building and competent genetic programming," in *GENETIC PROGRAMMING THEORY AND PRACTISE, CHAPTER 13*. Kluwer, 2003, pp. 205–220.

[23] G. Harik, "Linkage learning via probabilistic modeling in the ECGA," Tech. Rep., 1999.

[24] M. Looks, B. Goertzel, and C. Pennachin, "Learning computer programs with the Bayesian optimization algorithm," *Proceedings of the 2005 Genetic and evolutionary computation conference*, pp. 747 – 748, 2005.

[25] G. F. Cooper and E. Herskovits, "A Bayesian method for the induction of probabilistic networks from data," *Machine Learning*, vol. 9, pp. 309–347, 1992.

[26] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, pp. 461–464, 1978.

[27] T. Yanase, Y. Hasegawa, and H. Iba, "Binary encoding for prototype tree of probabilistic model building gp," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, ser. GECCO '09. New York, NY, USA: ACM, 2009, pp. 1147–1154. [Online]. Available: http://doi.acm.org/10.1145/1569901.1570055

[28] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[29] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *Int. J. Comput. Vision*, vol. 70, pp. 41–54, October 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1138215.1138220

[30] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings, "Decentralised coordination of low-power embedded devices using the max-sum algorithm," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 2*, ser. AAMAS '08. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2008, pp. 639–646. [Online]. Available: http://dl.acm.org/citation.cfm?id=1402298.1402313

[31] C. Gathercole, P. Ross, and S. Bridge, "An adverse interaction between the crossover operator and a restriction on tree depth," *in Proc. 1st Annu. Conf. Genetic Programming*, pp. 291–296, 1996.

[32] W. B. Langdon and R. Poli, "An analysis of the max problem in genetic programming," in *Advances in Genetic Programming 3, chapter 13*. MIT Press, 1997, pp. 301–323.

[33] W. F. Punch, "How effective are multiple populations in genetic programming," *Genetic Programming 1998: Proceedings of the Third Annual Conference: John R. Koza and Wolfgang Banzhaf and Kumar Chellapilla and Kalyanmoy Deb and Marco Dorigo and David B. Fogel and Max H. Garzon and David E. Goldberg and Hitoshi Iba and Rick Riolo, Eds*, pp. 308–313 308–313, 1998.

[34] M. Mitchell, S. Forrest, and J. H. Holland, "The royal road for genetic algorithms: Fitness landscapes and ga performance," *Proceedings of the First European Conference on Artificial Life*, pp. 245–254, 1992.

[35] K. Kim, B. McKay, and D. Punithan, "Sampling bias in estimation of distribution algorithms for genetic programming using prototype trees," in *Proceedings of the 11th Pacific Rim international conference on Trends in artificial intelligence*, ser. PRICAI'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 100–111. [Online]. Available: http://dl.acm.org/citation.cfm?id=1884293.1884307